Modelling Attacker's Knowledge for Cascade Cryptographic Protocols

Nazim Benaïssa *

Université Henri Poincaré Nancy 1
benaissa@loria.fr
LORIA
BP 239
54506 Vandœuvre-lès-Nancy
France

Abstract. We address the proof-based development of cryptographic protocols satisfying security properties. Communication channels are supposed to be unsafe. Analysing cryptographic protocols requires precise modelling of the attacker's knowledge. In this paper we use the event B modelling language to model the knowledge of the attacker for a class of cryptographic protocols called *cascade protocols*. The attacker's behaviour conforms to the Dolev-Yao model. In the Dolev-Yao model, the attacker has full control of the communication channel, and the cryptographic primitives are supposed to be perfect.

key-words: cryptography, model for attacker, formal methods

1 Introduction

Proving properties such as secrecy or authentication on cryptographic protocols is a crucial point. A protocol satisfies a secrecy property if it is able to prevent the attacker from learning the content of a secret message intended for other users. By authentication we mean that an attacker can not mislead other honest agents about his identity. To be able to prove such properties on a protocol, we must be able to model the knowledge of the attacker. One popular model of attacker's behaviour is the Dolev-Yao model [6]; this model is an informal description of all possible behaviours of the attacker as described in section 2. In this paper we present an event B [1, 2, 4] model of the attacker for a class of cryptographic protocols called *cascade protocols* and we prove the secrecy property on it. Our work is based on that of Dolev-Yao [6] where they gave a characterization of secure *cascade protocols*, but proofs in their work were done by hand.

Proving properties on cryptographic protocols such as secrecy is known to be undecidable. However research involving formal methods for the analysis of security protocols has been carried out. Theorem provers or model checkers are

* This work was supported by grant No. ANR-06-SETI-015-03 awarded by the Agence Nationale de la Recherche.

usually used for proving. For model checking, one famous example is Lowe's approach [7] using the process calculus CSP and the model checker FDR. Lowe discovered the famous bug in Needham-Schroeder's protocol. Model checking is efficient for discovering an attack if there is one, but it can not guarantee that a protocol is reliable. Many other works are based on theorem proving: Paulson [10] used an inductive approach to prove safety properties on protocols. He defined protocols as sets of traces and used the theorem prover Isabelle [9]. Other approaches, like Bolignano [3], combines theorem proving and model checking taking general formal method based techniques as a framework.

We summarize the organisation of the paper: in section 2, we present the Dolev-Yao attacker model. We then present the class of cascade protocols and the characterisation of secure protocols with respect to the secrecy property. The event B model of the attacker is given in section 3 of the paper.

2 The Dolev-Yao Model

In Dolev-Yao's model, cryptographic primitives are assumed to be black boxes satisfying given properties. The most important property is that the only way to obtain the plaintext M from the cipher text $K(M)$, where K is an encryption key, is to know the reverse key of K. In the Dolev Yao model, the attacker has full control of communication channels. He can intercept and remove any message from the channel, split unencrypted messages and decrypt parts of the message if he has the appropriate key. The attacker can also generate an infinite number of messages. All agents can be involved in an unlimited number of protocol instances, and interleaving of protocol instances have to be considered.

2.1 The Dolev-Yao Model for Cascade Protocols

Cascade protocols are a simple class of public protocols in which the agents involved in the protocol can apply several layers of encryption or decryption of messages. The encryption-decryption is made by using only public key operators. Dolev-Yao developed a model specifying the syntax of this class of protocols.

Let S be a set of symbols, we use S^* to denote the set of all finite sequences over S. Let E and D be respectively the set of encryptions and decryption keys of all the agents. If X is an agent, then his encryption key E_x and decryption key D_x are two functions mapping from $\{0,1\}^*$ into $\{0,1\}^*$. These functions satisfy the basic properties of the public key protocols: $E_x D_x = D_x E_x = id$, the identity function. D_x is known only by the agent himself while E_x is public and available in a key server.

Here is a short description of the Dolev-Yao model for cascade protocols (see [6] for detailed information). As shown in figure 1, a two party cascade protocol in the Dolev-Yao model is specified by a series of finite strings:

- $\alpha_i(X,Y) \in \{E_x, D_x, E_y\}^*, \quad 1 \leq i \leq t$

– $\beta_i(X,Y) \in \{E_y, D_y, E_x\}^*, \quad 1 \leq i \leq t' \; with \; t' = t \; or \; t-1$

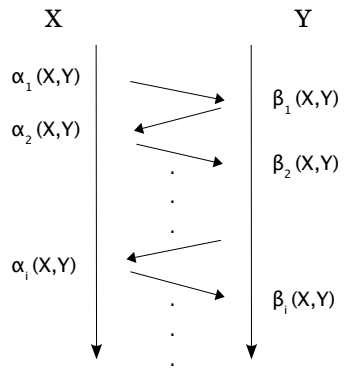

Fig. 1. A cascade protocol between two agents X,Y

When an agent X wishes to transmit a plaintext message M to another agent Y, the exchanged message has the following form: $N_k(X,Y)M$, where $1 \leq k \leq t + t'$ and:

– $N_1(X,Y) = \alpha_1(X,Y)$,
– $N_{2j}(X,Y) = \beta_j(X,Y)N_{2j-1}(X,Y), \quad 1 \leq j \leq t'$,
– $N_{2i+1}(X,Y) = \alpha_{i+1}(X,Y)N_{2i}(X,Y), \quad 1 \leq i \leq t-1$.

An attacker Z is supposed to be able to intercept any exchanged message between two agents X and Y, a cipher message $N_k(X,Y)M$ with $(k = 1, 2....)$, and will try to obtain the plaintext message M by applying different operators from one of three following categories:

1. $E \cup \{D_z\}$, E is known by all agents, and Dz is the attacker decryption key.
2. $\beta_i(X,Y)$ *for all* $X \neq Y$ *and* $i \geq 1$, even if the attacker does not know $\beta_i(X,Y)$'s value, he can start a transmission with any agent Y claiming himself to be agent X. He can then send any message *Msg* to Y in the $(2i-1)$st message and wait for Y's answer. He will then get $\beta_i(X,Y)$ applied to his message *Msg*.
3. $\alpha_i(X,Y)$ *for all* $X \neq Y$ *and* $i \geq 2$, in this case the attacker does not know the value of $\alpha_i(X,Y)$ but he may wait for X sending a message to Y, he can intercept Y's reply and prevent it from reaching X. He can then send any message to X claiming himself to be Y with his own message *Msg* and wait for the reply from X with $\alpha_i(A,B)$ applied to *Msg*.

As a result, the attacker will try to obtain the plaintext Message M from a cipher message $N_k(X,Y)M$ with $(k = 1, 2....)$ by applying operators from these three categories even if the he does not know the value of $\alpha_i(X,Y)$ or $\beta_i(X,Y)$ for two agents X and Y.

2.2 Secure Cascade Protocols in the Dolev-Yao Model

We give here two definitions from the Dolev-Yao model followed by the characterization of secure cascade protocol:

Definition 1. *Let $\pi \in (E \cup D)^*$ be a string and A be a user name. We say that π has the balancing property with respect to A if the following statement holds: if $D_A \in symb(\pi)$ then $E_A \in symb(\pi)$* [1]

Definition 2. *Let X,Y be two distinct user names. A two party cascade protocol is a balanced cascade protocol if*

1. *for every $i \geq 2$, $\alpha_i(X, Y)$ has the balancing property with respect to X, and*
2. *for every $j \geq 1$, $\beta_j(X, Y)$ has the balancing property with respect to Y.*

And the main result of the Dolev-Yao model is the following theorem.

Theorem 1. *Let X,Y be two distinct user names. A two-party cascade protocol is secure if and only if*

1. *$symb(\alpha_1(X, Y)) \cap \{E_x, E_y\} \neq \varnothing$, and*
2. *the protocol is balanced.*

After presenting the Dolev-Yao attacker model and the cascade protocols, we give in the next section an event B model of the attacker and prove on this model that if a cascade protocol is balanced then the secrecy property holds on this protocol.

3 Modelling the Attacker

First we give the static part of the model, the basic carrier sets are the following

- Msg: Set of all possible messages exchanged in the system.
- agent: Set of all agents including attackers.

We also define the set of encryption and decryption keys, respectively E and D. Two total injective functions EA, DA associate keys to their owners. Obviously, two different agents can not have the same encryption or decryption keys.

SETS

Msg
$agent$

AXIOMS

$axm1 : D \subseteq Msg \rightarrow Msg$
$axm2 : DA \in agent \rightarrowtail D$
$axm3 : E \subseteq Msg \rightarrow Msg$
$axm4 : EA \in agent \rightarrowtail E$

[1] $symb(\pi)$ is the set of symbols of π

The attacker is an agent among others, he has his own encryption and decryption key:

$$
\begin{array}{l}
axm5 : Z \in agent \\
axm6 : Dz \in D \\
axm7 : DA(Z) = Dz \\
axm8 : Ez \in E \\
axm9 : EA(Z) = Ez
\end{array}
$$

Cryptographic primitive are supposed to be perfect and only the decryption key of an agent can be used to decrypt a message encrypted with his encryption key, this is modeled by the use of sequences and the reduction operation over the sequences.

3.1 Key Sequences

In cascade protocols, agents may apply more than one key on a message they received. A possible modelling of encryptions where several keys are applied is the use of function composition. If X and Y are two agents, $(DA(X); EA(Y))(Msg)$ is an encryption with two keys $DA(X)$ and $EA(Y)$. The problem with using function composition is that it has no memory, and it is therefore not possible to write properties on the structure of an encryption with more than one key. Thus, we use sequences to model an encryption where several layers of keys are used. For example, if X and Y are two agents, $[EA(X), DA(Y), EA(X)]$ is an encryption sequence where $EA(X)$ is first applied, and is followed by $DA(Y)$ and $EA(X)$.

When an encryption key of an agent is immediately followed by a decryption key of the same agent in a sequence, this sequence can be reduced to a shorter sequence where both keys are removed. For example, if X and Y are two agents, $[EA(X), DA(X), EA(Y)]$ can be reduced to $[EA(Y)]$. Formally, we model the reduction relation as the smallest relation that satisfies:

$$
\begin{array}{l}
axm10 : reduction \in (\mathbb{N} \nrightarrow D \cup E) \leftrightarrow (\mathbb{N} \nrightarrow D \cup E) \\
axm11 : \forall seq1, seq2, i, j, k, A \cdot \\
\qquad A \in agent \land \\
\qquad i \mathinner{..} j \subseteq \mathbb{N} \land k \in i \mathinner{..} j \land k + 1 \in i \mathinner{..} j \land \\
\qquad seq1 \in i \mathinner{..} j \rightarrow D \cup E \land \\
\qquad seq1(k) = DA(A) \land seq1(k+1) = EA(A) \land \\
\qquad seq2 \in i \mathinner{..} j - 2 \rightarrow D \cup E \land \\
\qquad seq2 = i \mathinner{..} j - 2 \lhd (seq1 \lhd\!\!\!\!- \{l \mapsto m | l \in k \mathinner{..} j - 2 \land m = seq1(l+2)\}) \\
\qquad \Rightarrow \\
\qquad seq1 \mapsto seq2 \in reduction
\end{array}
$$

In the previous axiom, we considered the case of a decryption key followed by an encryption key. We added a similar axiom for the case where an encryption key is followed by a decryption key.

To guaranty that reductions are made only between the encryption and decryption key of the same agent, the injectivity of the functions DA and EA is not sufficient and it is necessary to be sure that an encryption key of an agent is not used as a decryption key of another agent.

$$axm12 : ran(EA) \cap ran(DA) = \varnothing$$

We emphasize that since we use the *reduction* relation, it is not necessary to have the following property on agents keys:

$$axm13 : \forall A \cdot A \in agent \Rightarrow (DA(A); EA(A)) = id(Msg)$$

It may be possible to apply several reductions iteratively over a sequence. Thus, the *reduction* relation needs to be applied iteratively. We use a relation *Rep* similar to the one used by Cansell and Méry in [5]. *Rep* behaves like a repeat-until loop, it captures the idea of repeating a relation on a set as long as it is possible to apply the relation. A pair $(seq1, seq2)$ is in *Rep* if either $seq1 \notin dom(reduction)$ and $seq1 = seq2$ or $seq1 \in dom(reduction)$ and there is a path over $reduction$ leading to $seq2 \notin dom(reduction)$. Formally, *Rep* is the smallest relation that satisfies:

$$axm14 : NotDOMAIN = id(\mathbb{N} \nrightarrow D \cup E) \setminus id(dom(reduction))$$
$$axm15 : Rep \in (\mathbb{N} \nrightarrow D \cup E) \leftrightarrow (\mathbb{N} \nrightarrow D \cup E)$$
$$axm16 : Rep = NotDOMAIN \cup (reduction; Rep)$$

When no more reductions are possible, we say that the sequence is in the *normal form*. Formally, the normal form is modeled as follows:

$$axm17 : Norm \in ((\mathbb{N} \nrightarrow D \cup E) \rightarrow (\mathbb{N} \nrightarrow D \cup E))$$
$$axm18 : Norm \subseteq Rep$$

If the normal form of a sequence *seq* equals the empty set, it means that the composition of all encryption and decryption keys contained in the sequence equals the identity function and we can obtain the plain text M from $seqM$.

seq_Ai and seq_Bj are two sets containing sequences of encryption or decryption keys. If X and Y are two agents involved in a protocol run, seq_Ai contains all sequences of keys applied in each step of the protocol by agent X, seq_Bj contains those applied by Y. Each sequence contained in one of these sets matches with an $\alpha_i(X, Y)$ or $\beta_j(X, Y)$ defined in the Dolev-Yao model.

$$
\begin{array}{l}
axm19 : seq\_Ai \subseteq \mathbb{N} \pfun D \cup E \\
axm20 : seq\_Bj \subseteq \mathbb{N} \pfun D \cup E \\
axm21 : \forall seq \cdot seq \in (seq\_Ai \cup seq\_Bj) \\
\qquad \Rightarrow \\
\qquad (\\
\qquad\quad \exists X, Y \cdot X \in agent \land Y \in agent \land \\
\qquad\quad X \neq Y \land \\
\qquad\quad ran(seq) \subseteq \{DA(X), EA(X), EA(Y)\} \\
\qquad)
\end{array}
$$

The protocol has to be *balanced* (see definition 2), thus for each sequence from the sets seq_Ai and seq_Bj the following axioms holds:

$$
\begin{array}{l}
axm22 : \forall X, seq \cdot X \in agent \land seq \in seq\_Ai \land \\
\qquad\qquad DA(X) \in ran(seq) \Rightarrow EA(X) \in ran(seq) \\
axm23 : \forall Y, seq \cdot Y \in agent \land seq \in seq\_Bj \land \\
\qquad\qquad DA(Y) \in ran(seq) \Rightarrow EA(Y) \in ran(seq)
\end{array}
$$

We emphasize the particular case of the first step of the protocol that is not concerned by the previous axiom22. We define a set seq_A1 containing the sequences corresponding to the first step of the protocol. It is not mandatory for sequences from this set to satisfy the *balancing property*, but they should at least contain one encryption key as stated in the Dolev-Yao characterization of secure protocols (see theorem 1):

$$
\begin{array}{l}
axm24 : seq\_A1 \subseteq \mathbb{N} \pfun D \cup E \\
axm25 : \forall seq \cdot seq \in seq\_A1 \\
\qquad \Rightarrow \\
\qquad (\\
\qquad\quad \exists X, Y \cdot X \in agent \land Y \in agent \land \\
\qquad\quad X \neq Y \land \\
\qquad\quad ran(seq) \subseteq \{DA(X), EA(X), EA(Y)\} \land \\
\qquad\quad ran(seq) \cap \{EA(X), EA(Y)\} \neq \varnothing \\
\qquad)
\end{array}
$$

3.2 Variables

We use a variable seq_Atk to model the structure of the messages that the attacker can obtain through applying his own keys or applying different sequences from the sets seq_Ai and seq_Bj. We also use a variable $size$ containing the size of the sequence seq_Atk and a variable $a1$ that memorizes the size of the sequence from the set seq_A1 used in the first step of the current protocol instance.

VARIABLES
seq_Atk
$size$
$a1$

INVARIANTS
$inv1 : size \in \mathbb{N}_1$
$inv2 : a1 \in \mathbb{N}_1$
$inv3 : seq\_Atk \in 1 \mathinner{..} size \rightarrow D \cup E$

We emphasize that the variable seq_Atk does not contain the plain text message M, but only the sequence of public key operators that may be applied by the attacker. Thus, in order to prove that the protocol satisfies the secrecy property, we must prove that the normal form of the sequence seq_Atk is never equal to the empty set. If the normal form of a sequence equals the empty set, it means that the composition of all encryption and decryption keys contained in the sequence equals the identity function and the attacker can obtain the plaintext M.

$thm2 : Norm(seq\_Atk) \neq \varnothing$

3.3 Events

The attacker can intercept any message exchanged between two agents. When a honest agent initiates a transaction with another agent, he first applies to the plain text message M a sequence from the set seq_A1 (first step of the protocol). The two agents apply then alternately sequences from seq_Ai and seq_Bj. Messages exchanged between agents have the form "$(seq\_Ai \cup seq\_Bj)^*$ seq_A1 M", M is the plaintext message. After intercepting the cipher message "$(seq\_Ai \cup seq\_Bj)^*$ seq_A1 M", the attacker applies different sequences from the set $seq\_Ai \cup seq\_Bj \cup E \cup \{D_z\}$. Accordingly, there is no need to model explicit message interception by the attacker, it is enough to initialize the variable seq_Atk with a sequence from the set seq_A1 and add events that model the concatenation of seq_Atk with all possible sequences:

- Initialization of seq_Atk with a sequence from seq_A1.
- Event $Attack\_seq\_Ai$: concatenation of seq_Atk with a sequence from seq_Ai.
- Event $Attack\_seq\_Bj$: concatenation of seq_Atk with a sequence from seq_Bj.
- Event $Attack\_E$: concatenation of seq_Atk with a sequence from E.
- Event $Attack\_D_z$: concatenation of seq_Atk with D_z.

These concatenations are done by some honest agent before the message is intercepted or by the attacker himself after intercepting the cipher message.

In order to write the appropriate events, we need to have tools that let us manipulate sequences such as concatenation or subsequences. In our model we use a modified form of the relation *match* introduced by Jean Raymond Abrial in the Earley algorithm model. We modified this relation to adapt it to our case study:

$axm26 : match \in (\mathbb{N} \pfun D \cup E) \leftrightarrow (\mathbb{N} \pfun D \cup E)$
$axm27 : \varnothing \mapsto \varnothing \in match$

Unlike the equality, two sequences $seq1 \in i..j \rightarrow D \cup E$ and $seq2 : k..l \rightarrow D \cup E$ may match if the order of the keys in the two sequences is the same even if their respective domains $i..j$ and $k..l$ are different (see example in figure 2).

$$
\begin{array}{l}
axm28 : \forall i, j, k, l, n1, n2, s1, s2 \cdot \\
\quad i \in 1 \mathinner{..} j + 1 \land \\
\quad j \in 0 \mathinner{..} n1 - 1 \land \\
\quad k \in 1 \mathinner{..} l + 1 \land \\
\quad l \in 0 \mathinner{..} n2 - 1 \land \\
\quad s1 \in 1 \mathinner{..} n1 \rightarrow D \cup E \land \\
\quad s2 \in 1 \mathinner{..} n2 \rightarrow D \cup E \land \\
\quad i \mathinner{..} j \lhd s1 \mapsto k \mathinner{..} l \lhd s2 \in match \land \\
\quad s1(j+1) = s2(l+1) \\
\quad \Rightarrow \\
\quad i \mathinner{..} j + 1 \lhd s1 \mapsto k \mathinner{..} l + 1 \lhd s2 \in match
\end{array}
$$

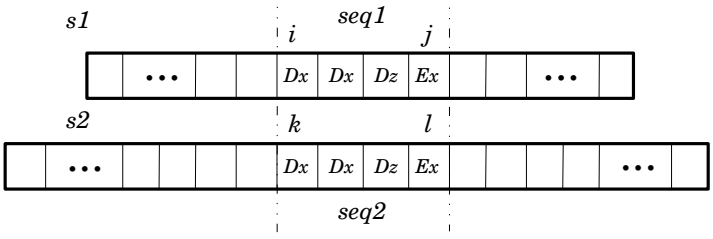

Fig. 2. The match relation

We also add a fixed point axiom saying that *match* is the smallest relation satisfying the axiom 28. Using *match* is convenient to express relations between sequences. For instance, to express the fact that a sequence *seq1* is a subsequence of *seq2*, it suffices to say that there are some i, j such that $seq1 \mapsto i \mathinner{..} j \lhd seq2 \in match$. To express the fact that a sequence $seq \in i..j \rightarrow D \cup E$ is the result of the concatenation of two sequences *seq1* and *seq2*, it suffices to say that there is some k such that $seq1 \mapsto i \mathinner{..} k \lhd seq \in match$ and $seq2 \mapsto k+1 \mathinner{..} j \lhd seq \in match$.

Events have been added to the model to express all the attacker's options. The following event shows the case of a sequence randomly chosen from the set *seq_Ai*. This sequence is concatenated with the attacker sequence *seq_Atk*, the variable size is also increased.

EVENT sendAi
ANY
seq_Ax
ax
WHERE
$grd1 : seq\_Ax \in seq\_Ai$
$grd2 : ax \in \mathbb{N}_1$
$grd3 : seq\_Ax \in 1 \mathinner{..} ax \rightarrow D \cup E$
THEN
$act1 : size := size + ax$
$act2 : seq\_Atk : | \; seq\_Atk' \in 1 \mathinner{..} size + ax \rightarrow D \cup E \wedge$
$seq\_Ax \mapsto 1 \mathinner{..} ax \lhd seq\_Atk' \in match \wedge$
$seq\_Atk \mapsto ax + 1 \mathinner{..} ax + size \lhd seq\_Atk' \in match$
END

Similar events are added to express all the other possibilities of the Dolev-Yao model. Since the attacker sequence is initialized with a sequence from the set seq_A1, it will have two parts (as shown in figure 3). A part $1..size-a1 \lhd seq\_Atk$ that matches with a sequence from $(seq\_Ai \cup seq\_Bj \cup E \cup \{D_z\})^*$, and a part $(size-a1)+1 \mathinner{..} size \lhd seq\_Atk$ that matches with a sequence from seq_A1. It's important to distinguish these two parts since, unlike sequences from the set $seq\_Ai \cup seq\_Bj$, sequences from the set seq_A1 do not satisfy the *balancing property.*

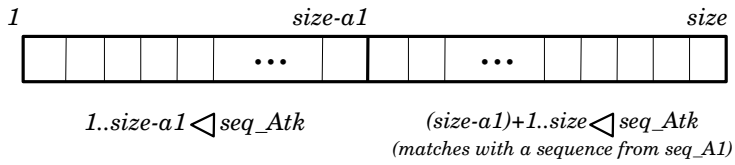

Fig. 3. The two parts of the attacker sequence

3.4 Invariant and Proofs

Proofs of the B model are inspired from the proofs given by Dolev and Yao in their model, but proofs of their models were done by hand and parts of their proofs were stated without being formally proved. Before introducing the main invariant of our model we first give definitions of some important properties over sequences that are necessary to state the invariant. $A\_Norm(A)(seq)$ is the

normal form with respect to one agent A of a sequence seq, it is obtained by removing all possible subsequences $[EA(A), DA(A)]$ or $[DA(A), EA(A)]$.

$$axm29 : A\_Norm \in agent \rightarrow ((\mathbb{N} \nrightarrow D \cup E) \rightarrow (\mathbb{N} \nrightarrow D \cup E))$$

For example,

$$A\_Norm(X)([DA(Y), EA(Y), EA(X), DA(X)]) = [DA(Y), EA(Y)]$$

We modeled A_Norm similarly to $Norm$ using a reduction relation where only keys from the appropriate agent are reduced.

In order to prove that the normal form of the attacker sequence never equals the empty set, we need to prove first that the sequence $Norm(1 \mathinner{..} size - a1 \lhd seq\_Atk)$ has the *balancing property* with respect to all agents except the attacker himself. We recall that it is not mandatory to have the *balancing property* for sequences from the set seq_A1, this is why this property does not hold for the whole attacker sequence.

$$\begin{array}{l} thm3 : \forall A \cdot A \in agent \land A \neq Z \land \\ \quad DA(A) \in ran(Norm(1 \mathinner{..} size - a1 \lhd seq\_Atk)) \\ \quad \Rightarrow \\ \quad EA(A) \in ran(Norm(1 \mathinner{..} size - a1 \lhd seq\_Atk)) \end{array}$$

But unfortunately this property is not an inductive invariant but only a theorem. As a counter example, let us consider the case where $Norm(1 \mathinner{..} size - a1 \lhd seq\_Atk)$ equals:

$$[DA(A), DA(Z), EA(A), EA(Z), DA(X), EA(Y), DA(A), DA(Y), EA(X)]$$

This sequence satisfies the balancing property. If the previous event $sendAi$ is triggered with the local variable $seq\_Ax = [DA(A), EA(Z), EA(A)]$ (this sequence satisfies the axioms 19 and 21), the new value of $Norm(1 \mathinner{..} size - a1 \lhd seq\_Atk)$ will be: $[EA(Z), DA(X), EA(Y), DA(A), DA(Y), EA(X)]$. The new value does not satisfy the balancing property anymore. Thus we introduce a new property called *A_Balanced property* of a sequence with respect to an agent A:

$$\begin{array}{l} axm30 : A\_Balanced \in agent \rightarrow \mathbb{P}(\mathbb{N} \nrightarrow D \cup E) \\ axm31 : \forall A, seq, i, j \cdot seq \in i \mathinner{..} j \rightarrow D \cup E \land \\ \quad i \mathinner{..} j \subseteq \mathbb{N} \land j \geq i \land \\ \quad (seq(i) \in D \setminus \{DA(A)\} \land \\ \quad seq(j) \in D \setminus \{DA(A)\} \land \\ \quad ran((A\_Norm(A))(i + 1 \mathinner{..} j - 1 \lhd seq)) \cap D \subseteq \{DA(A)\} \land \\ \quad DA(A) \in ran((A\_Norm(A))(i + 1 \mathinner{..} j - 1 \lhd seq)) \Rightarrow \\ \quad\quad EA(A) \in ran((A\_Norm(A))(i + 1 \mathinner{..} j - 1 \lhd seq))) \\ \quad \Rightarrow \\ \quad seq \in A\_Balanced(A) \end{array}$$

Intuitively, for an agent A, a sequence is $A\_Balanced(A)$ means that if the first and last symbols of this sequence are decryption keys and if the $A\_Norm(A)$ of this sequence contains only A decryption key in its range it should also contain A encryption key.
The main invariant of our model states that each subsequence of the sequence $D_z\ (1\,..\,size - a1 \lhd seq\_Atk)\ D_z$ has the *A_Balanced property* with respect to all agents except the attacker.

$$
\begin{array}{l}
inv4 : \forall seq, i, j, k, l, A, seq\_Atk\_Dz \cdot \\
\quad A \in agent \wedge A \neq Z \wedge \\
\quad seq \in i\,..\,j \rightarrow D \cup E \wedge \\
\quad seq\_Atk\_Dz \in 1\,..\,size - a1 + 2 \rightarrow D \cup E \\
\quad seq\_Atk\_Dz(1) = Dz \\
\quad seq\_Atk\_Dz(size - a1 + 2) = Dz \\
\quad 1 \ldots size - a1 \lhd seq\_Atk \mapsto 2\,..\,size - a1 + 1 \lhd seq\_Atk\_Dz \in match \\
\quad seq \mapsto k\,..\,l \lhd seq\_Atk\_Dz \in match \\
\quad \Rightarrow \\
\quad seq \in A\_Balanced(A)
\end{array}
$$

Let us consider the case of the event *sendAi* shown before. In this event, we concatenate a sequence from the set *seq_Ai* to the sequence *seq_Atk* to obtain *seq_Atk'* . We have then to prove that any subsequence *seq* of $D_z\ (1\,..\,size - a1 \lhd seq\_Atk)\ D_z$ (there are some k, l such that $seq \mapsto k\,..\,l \lhd (D_z\ (1\,..\,size - a1 \lhd seq\_Atk')\ D_z) \in match$) is *A_Balanced* (see figure 4).

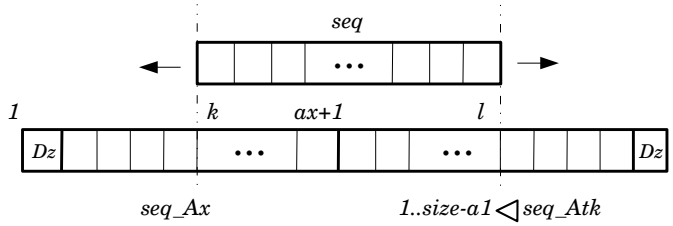

Fig. 4. A_Balanced property has to be proved on seq

To achieve the proof, all possible cases of $k..l$ values have to be considered (especially the values of k and l compared to the value of $ax + 1$), this is made easier by the use of the *match* relation. For each case it is necessary to prove that the concatenation of a sequence that has the *balancing property* with respect to an agent A with a sequence that has the *A_Balanced property* with respect to A results on a sequence that has the *A_Balanced property* with respect to A, since this has to be done with all events of the model, it was interesting to prove the following theorem:

$$
\begin{array}{l}
thm4 : \forall seq, i, j, k, n, A \cdot \\
\quad A \in agent \land seq \in 1 \mathinner{..} n \rightarrow D \cup E \land \\
\quad i \mathinner{..} j \subseteq 1 \mathinner{..} n \land k \in i \mathinner{..} j \land \\
\quad A\_Norm(A)(i \mathinner{..} k \lhd seq) = i \mathinner{..} k \lhd seq \land \\
\quad (DA(A) \in ran(i \mathinner{..} k \lhd seq) \Rightarrow EA(A) \in ran(i \mathinner{..} k \lhd seq)) \land \\
\quad (DA(A) \in ran(A\_Norm(A)(k+1 \mathinner{..} j \lhd seq)) \Rightarrow \\
\qquad\qquad EA(A) \in ran(A\_Norm(A)(k+1 \mathinner{..} j \lhd seq))) \\
\quad \Rightarrow \\
\quad (DA(A) \in ran(A\_Norm(A)(i \mathinner{..} j \lhd seq)) \Rightarrow \\
\qquad\qquad EA(A) \in ran(A\_Norm(A)(i \mathinner{..} j \lhd seq)))
\end{array}
$$

The last step of our modelling is to prove theorem 2, that states that seq_Atk never equals the empty set, from the theorem 3 that states that the sequence $Norm(1 \mathinner{..} size - a1 \lhd seq\_Atk)$ has the *balancing property* with respect to all agents other than the attacker. To prove this result, we do a proof by case on the structure of $seq = size - a1 + 1 \mathinner{..} size \lhd seq\_Atk$ (the part of the attacker sequence that matches with the first step of the protocol). According to axiom 23, there are two agents X,Y such that $ran(seq) \subseteq \{DA(X), EA(X), EA(Y)\}$ and $ran(seq) \cap \{EA(X), EA(Y)\} \neq \varnothing$. We give here a sketch of the proof:
By contradiction, suppose that the normal form of seq_Atk equals the empty set, two case are possible:

1. $EA(Y) \in ran(seq)$: since $DA(Y) \notin ran(seq)$, the only way to obtain the empty set in the normal form of the whole sequence seq_Atk is that the reminder part $Norm(1 \mathinner{..} size - a1 \lhd seq\_Atk)$ contains $DA(Y)$ but not $EA(Y)$. This is impossible because of the *balancing property* of this part of the attacker sequence.
2. The other case is done in a similar way.

Proving these invariants and theorems requires intensive use of operators over sequences. The axiom defining the relation *match* given before is not convenient in our case, that's why we introduced several theorems over this relation such as identity, reflexivity and transitivity properties to make proofs easier. Here follows an example of one of these theorems:

$$
\begin{array}{l}
thm5 : \forall seq1, seq2, k, i1, i2, j1, j2 \cdot \\
\quad seq1 \in i1 \mathinner{..} j1 \rightarrow D \cup E \land \\
\quad seq2 \in i2 \mathinner{..} j2 \rightarrow D \cup E \land \\
\quad k \in 0 \mathinner{..} j1 - i1 \land \\
\quad seq1 \neq \varnothing \land seq2 \neq \varnothing \land \\
\quad seq1 \mapsto seq2 \in match \\
\quad \Rightarrow \\
\quad seq1(i1 + k) = seq2(i2 + k)
\end{array}
$$

To prove this theorem we used induction over the size of the sequence. These theorems are not specific to this model, thus they can be reused later in similar

protocol models. We used the Rodin platform [8] for modelling and proving our attacker model. 10 theorems were proved interactively on the *match* relation. 25 proofs generated by the prover for the invariants of the model, 13 were done automatically. Interactive proofs were not difficult except proofs of the main invariant 4 of the model that were long because of the high number of the cases that had to be considered.

4 Conclusion

We have written in this paper a B model of the attacker for a class of cryptographic protocols. Events of our model take into account all the options the attacker can perform in the Dolev-Yao model. Unlike the original Dolev-Yao's model for cascade protocols, proofs were mechanized. Accordingly, all constraints on the attacker's model have to be stated explicitly, and some of the constraints were added later during the proving process. Proofs of our model were made easier by the use of the *match* relation and by the theorems we have proved over this relation. These theorems can be reused in future developments. The next step will be modelling attackers for more complex classes of protocols and to study how attacker models can be integrated into the complete protocol model.

Acknowledgements Thanks are due to Jean Raymond Abrial for his advice on modelling cryptographic protocols. We also thank Dominique Cansell and Dominique Méry for their help and suggestions.

References

1. J.-R. Abrial. *The B book - Assigning Programs to Meanings*. Cambridge University Press, 1996.
2. Dines Bjørner and Martin C. Henson, editors. *Logics of Specification Languages*. EATCS Textbook in Computer Science. Springer, 2007.
3. Dominique Bolignano. Integrating proof-based and model-checking techniques for the formal verification of cryptographic protocols. In *CAV*, pages 77–87, 1998.
4. Dominique Cansell and Dominique Méry. *The event-B Modelling Method: Concepts and Case Studies*, pages 33–140. Springer, 2007. See [2].
5. Dominique Cansell and Dominique Méry. Incremental parametric development of greedy algorithms. *Electr. Notes Theor. Comput. Sci.*, 185:47–62, 2007.
6. D. Dolev and A. Yao. On the security of public key protocols. *Information Theory, IEEE Transactions on*, 29(2):198–208, Mar 1983.
7. Gavin Lowe. Breaking and fixing the needham-schroeder public-key protocol using fdr. In *TACAS*, pages 147–166, 1996.
8. Christophe Metayer, Jean-Raymond Abrial, and Laurent Voisin. Event-B language. RODIN Project Deliverable D7, May 2005.
9. Lawrence C. Paulson. *Isabelle - A Generic Theorem Prover (with a contribution by T. Nipkow)*, volume 828 of *Lecture Notes in Computer Science*. Springer, 1994.
10. Lawrence C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6:85–128, 1998.